Fingerprinting Cloud FPGA Infrastructures

Shanquan Tian
Yale University
shanquan.tian@yale.edu

Wenjie Xiong
Yale University
wenjie.xiong@yale.edu

Ilias Giechaskiel
University of Oxford
ilias.giechaskiel@cs.ox.ac.uk

Kasper Rasmussen
University of Oxford
kasper.rasmussen@cs.ox.ac.uk

Jakub Szefer
Yale University
jakub.szefer@yale.edu

ABSTRACT

In recent years, multiple public cloud FPGA providers have emerged, increasing interest in FPGA acceleration of cryptographic, bioinformatic, financial, and machine learning algorithms. To help understand the security of the cloud FPGA infrastructures, this paper focuses on a fundamental question of understanding what an adversary can learn about the cloud FPGA infrastructure itself, without attacking it or damaging it. In particular, this work explores how unique features of FPGAs can be exploited to instantiate Physical Unclonable Functions (PUFs) that can distinguish between otherwise-identical FPGA boards. This paper specifically introduces the first method for identifying cloud FPGA instances by extracting a unique and stable FPGA fingerprint based on PUFs measured from the FPGA boards' DRAM modules. Experiments conducted on the Amazon Web Services (AWS) cloud reveal the probability of renting the same physical board more than once. Moreover, the experimental results show that hardware is not shared among `f1.2xlarge`, `f1.4xlarge`, and `f1.16xlarge` instance types. As the approach used does not violate any restrictions currently placed by Amazon, this paper also presents a set of defense mechanisms that can be added to existing countermeasures to mitigate users' attempts to fingerprint cloud FPGA infrastructures.

CCS CONCEPTS

• **Security and privacy** → *Hardware attacks and countermeasures*;
• **Hardware** → *Reconfigurable logic and FPGAs*;

KEYWORDS

Cloud FPGAs, Fingerprinting, Physical Unclonable Functions, PUFs, DRAM PUFs, Data Retention, DRAM Refresh, DRAM Decay

ACM Reference Format:
Shanquan Tian, Wenjie Xiong, Ilias Giechaskiel, Kasper Rasmussen, and Jakub Szefer. 2020. Fingerprinting Cloud FPGA Infrastructures. In *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '20), February 23–25, 2020, Seaside, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3373087.3375322

FPGA '20, February 23–25, 2020, Seaside, CA, USA

ACM ISBN 978-1-4503-7099-8/20/02…$15.00
https://doi.org/10.1145/3373087.3375322

1 INTRODUCTION

The proliferation of cloud FPGA infrastructures has made on-demand access to FPGA acceleration available for several applications, including financial modeling, cryptography, and genome data analysis, among others [5]. The wide availability of FPGAs has many benefits, but the potentially highly-sensitive nature of the information processed has attracted recent research on FPGA covert-channel attacks. Specifically, multi-tenant [16, 17] and temporal [38] covert communication was shown to be possible in cloud FPGAs, despite some countermeasures deployed by the cloud providers [9].

Such attacks, however, make a crucial assumption in their threat model, namely that the adversary has some knowledge of the cloud FPGA infrastructure itself. In other words, it is assumed that attackers know that their designs are co-located with the victim logic on the same FPGA chip (for multi-tenant attacks), or that the victim had rented the same physical FPGA board as the attacker in the previous time slot (for temporal attacks). Rather than focusing on attacks, this work focuses on the assumption they make, and shows for the first time that it is indeed possible to fingerprint the cloud infrastructure to deduce, for example, that the same FPGA chip has been reused in consecutive Virtual Machine (VM) allocations.

Existing cloud FPGA providers, such as Amazon Web Services (AWS) [11], secure their infrastructures through a number of measures. The architectures are not disclosed publicly, except for the types of FPGA chips used and the geographic location of the data centers. Furthermore, there are limitations on the designs that can be deployed in the cloud FPGAs. The AWS workflow, for example, shown in Figure 1, performs a number of Design Rule Checks (DRCs) on the Design Checkpoint (DCP) files generated by Xilinx's Vivado tools before the generated bitstream (called an Amazon FPGA Image, or AFI) can be loaded onto one of the AWS FPGAs. The checks, which include prohibiting combinatorial loops [9], are combined with a restrictive "shell" interface that prevents access to Xilinx eFUSE and Device DNA primitives [41], which could be used to identify the specific FPGA hardware that a user has rented.

In spite of the efforts to hide information about the cloud FPGA architecture, this paper shows that it is possible to get insights into the infrastructure through the resources that are available to unprivileged FPGA users. Specifically, this paper introduces the first algorithm for fingerprinting cloud FPGAs through unique features in their boards. Our approach uses Physical Unclonable Functions (PUFs) based on the decay of Dynamic Random Access Memory (DRAM) [43] to identify the DRAM modules attached to the cloud FPGA boards, and, by extension, the FPGAs themselves.

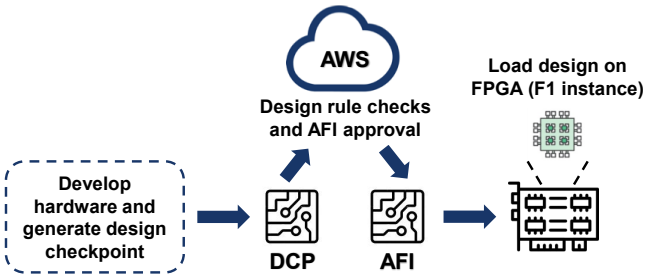

Figure 1: FPGA workflow on AWS: developers compile their custom logic (locally, or on any AWS node) and send encrypted Design Checkpoints (DCPs) to Amazon. DCPs which pass Design Rule Checks (DRCs) generate bitstream files that can be loaded onto F1 instances in the form of Amazon FPGA Images (AFIs).

To realize the fingerprinting, a novel approach had to be developed for instantiating decay-based DRAM PUFs. These PUFs require disabling the DRAM refresh commands to expose manufacturing variations [43], but the DRAM controller in cloud FPGAs is a black-box IP module from Xilinx [4], with no ability to control the refresh rate. As a result, direct access to decay-based DRAM PUFs is not possible. The FPGA DRAMs, however, are not erased when the same user loads a new design (AFI image):[1] this is part of a data-retention feature aimed at sharing data between different AFIs [3]. For the feature to work, consecutively-loaded AFIs instantiate DRAM controllers so that data is not lost. However, in our work, one AFI does not have DRAM controllers at all. By loading the AFIs with and without DRAM controllers instantiated, refresh of the DRAM modules is effectively disabled: AFIs without DRAM controllers keep DRAMs powered, but provide no refresh signals, which results in DRAM cells decaying, as needed by the PUFs.

Through this novel approach for implementing decay-based DRAM PUFs on cloud FPGAs, our work shows that it is possible to fingerprint the AWS F1 infrastructure and to build a profile of `f1.2xlarge`, `f1.4xlarge`, and `f1.16xlarge` instances.

Contributions

The contributions of this paper are as follows:

(1) After describing the relevant background (Section 2), we introduce a novel experimental setup which uses DRAM PUFs to fingerprint AWS cloud FPGAs (Section 3).
(2) We conduct the first fingerprinting experiments on cloud FPGAs, extracting unique and stable fingerprints of several Amazon `f1.2xlarge`, `f1.4xlarge` and `f1.16xlarge` instances (Section 4). Our evaluation is the first to show that there is no overlap between FPGAs of different instance types. We also calculate the probability of renting the same FPGA as a function of time, and demonstrate that DRAM PUFs can monitor changes in the data center temperature.
(3) We propose a set of countermeasures against cloud FPGA fingerprinting (Section 5).

We finally place our work in the context of cloud attacks and defenses (Section 6) before concluding (Section 7). The software scripts for data collection and analysis, as well as pre-compiled Amazon FPGA Images (i.e., AFI bitstreams) will be made available at https://caslab.csl.yale.edu/code/cloud-fpga-fingerprinting.

[1] The DRAMs are cleared when a *new* user is assigned to the FPGA instance.

2 BACKGROUND

This section describes current public cloud FPGA deployments (Section 2.1) and their typical hardware setup (Section 2.2). It also summarizes decay-based DRAM PUFs (Section 2.3), and states the threat model for the fingerprinting work (Section 2.4).

2.1 Cloud FPGAs

Several options are available for renting FPGAs in the cloud. Since 2015, academic researchers can access a cluster with Intel Stratix V FPGAs in the Texas Advanced Computing Center (TACC) [37]. Intel FPGAs are also available on Alibaba Cloud [1] and on Microsoft Azure for machine learning applications [25]. Xilinx-based cloud offerings have been available since 2016, when AWS announced F1 instances with Xilinx Virtex UltraScale+ FPGAs [2]. The same chips also power Huawei [40] and Alibaba [1] cloud services. Meanwhile, Kintex UltraScale boards are available in beta on Baidu [12] and Tencent [36], while Nimbix is equipped with Alveo cards [26].

2.2 Typical Cloud FPGA Setup

In a typical cloud FPGA deployment, a set of FPGA boards is connected to a server over PCIe. The boards contain FPGA chips, and are placed in fixed slots in the server. Each FPGA has access to four dedicated DRAM modules. As the fingerprinting results of our work show (Section 4), the four DRAM modules always appear in the same order within a given FPGA. Moreover, for each instance type (2x, 4x, and 16x), the same set of FPGAs is always rented together. In other words, there is no randomization or other dynamic change to the hardware setup: the set and order of FPGAs in a server remains constant, except in cases of hardware failure, or when FPGAs are added or removed from the data center.

2.3 Decay-Based DRAM PUFs

Dynamic Random-Access Memory (DRAM) is widely used in personal computers and servers due to its high storage density. Usually, multiple DRAM chips (*ranks*) are combined in a DRAM module to provide enough memory. Each DRAM chip consists of DRAM *banks*, which are arrays of DRAM cells, as shown in the "DRAM schematic" part of Figure 2. A single DRAM cell consists of a capacitor and a transistor, with bits of information stored as charges on the capacitors. The gate of the access transistor in the DRAM cell connects to the *wordline* (WL) in that row, while the capacitor in the DRAM cell connects to the *bitline* (BL) through the transistor. To access a certain memory address, the bitlines are first reset by the *equalizers*. Then, the corresponding wordline is enabled, and the charge on the capacitors is read through the *sense amplifiers*.

DRAM is a type of *volatile* memory, because the capacitor charge leaks over time through different leakage paths, as shown in Figure 2. The time that a DRAM cell can retain the charge on the capacitor and store the data value is called the *retention time*. After the retention time elapses, the charge on the cell will leak, and the bit stored in the DRAM cell may flip its value. To maintain the data integrity of information stored, the DRAM is refreshed periodically

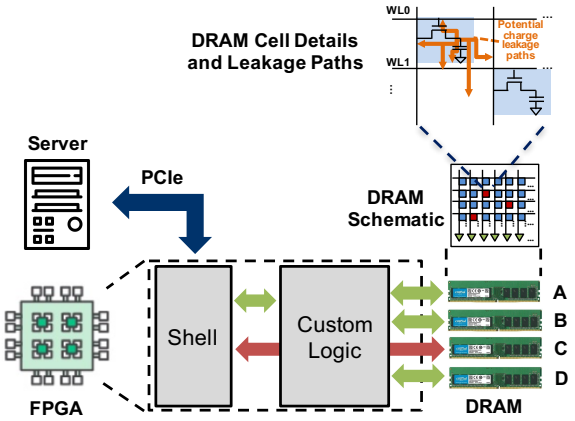

Figure 2: System diagram: a virtual machine communicates with one or more FPGAs over PCIe. Of the four DRAM modules on each FPGA board, one (DRAM C) is reserved by the shell. PUFs which exploit DRAM charge leakage on the other three DRAMs (A, B, and D) can uniquely identify the underlying hardware.

to recharge the capacitors to their original voltage levels. Moreover, an Error-Correcting Code (ECC) can also be applied.

The variation in the retention time of different DRAM cells can be used in Physical Unclonable Functions (PUFs) [43]. Specifically, in a decay-based DRAM PUF, the DRAM PUF region is first set to a known *initial value* (e.g., all ones) and the DRAM refresh is disabled. After a certain *decay period* elapses, the DRAM PUF region is then read. Due to DRAM charge leakage, bit flips (errors) in the initial values will occur. The location of the bit flips depends on variations in the fabrication process, and is considered to be unique for each DRAM chip. Thus, the bit flips due to DRAM decay can be used as a PUF response. DRAM PUFs have been used to identify and authenticate DRAM chips [27, 28, 30, 31, 33, 43], or generate keys [29, 31, 33, 43]. In this paper, we use DRAM data retention properties to create a unique fingerprint of DRAM chips, and, by extension, the FPGAs to which they are attached.

2.4 Threat Model

Covert- and side-channel attacks are possible in cloud FPGAs (Section 6), but they often require that adversaries be able to uniquely identify FPGA instances to carry out the attacks. This work provides a way to uniquely fingerprint individual FPGAs, while obeying design rules imposed by cloud FPGA providers. It is therefore assumed that the fingerprinting designs do not contain any prohibited circuits, such as combinatorial loops [9]. In addition, users only interact with the physical interfaces through the cloud-provided IP modules, such as the DRAM controller. Finally, attackers are not able to decrypt the protected IP, or otherwise reverse-engineer it to change its functionality.

3 FINGERPRINTING SETUP

This section explains the FPGA fingerprinting setup. Specifically, it presents a novel way to create decay-based DRAM PUFs by loading and unloading two different types of AFIs: one with and one without a memory controller. This approach disables refresh, while still providing power to the DRAM modules. Section 3.1 expands

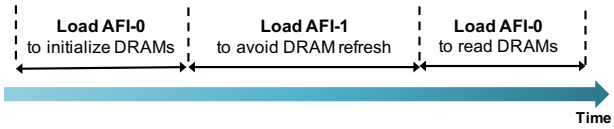

Figure 3: Steps to measure DRAM PUFs: `AFI-0` is first loaded to write all 1s to a certain area of a DRAM module. Then `AFI-1` is loaded to stop memory self-refresh. Finally, after a fixed amount of time, `AFI-0` is re-loaded to measure bit flips in the written addresses.

on the memory-related aspects of our experimental setup, while Section 3.2 explains in detail how DRAM PUFs are instantiated and used for data collection on AWS F1 instances.

3.1 Accessing DRAM from the FPGA

The `cl_dram_dma` example in the AWS development kit [11] explains how to access the DRAM from the FPGA. The physical pinout and timing parameters of the DDR4 DRAM chips are hidden in the `sh_ddr` module, which only provides a 512-bit AXI4 interface to user logic. It also implements memory initialization, error correction, and self-refresh of DRAM cells. As shown in Figure 2, although there are four DRAM modules, one (DRAM C) is reserved by the FPGA "shell". It is always initialized (and refreshed) regardless of whether custom user logic has instantiated a DRAM controller to use the memory. The remaining DRAMs (A, B, and D) are instantiated within the custom logic [4]. Each instantiation also uses the `sh_ddr` module, which is encrypted and prevents users from modifying its functionality: self-refresh of DRAM cells is always enabled whenever the DRAM controller is instantiated. Nevertheless, Section 3.2 presents a novel approach through a method that disables self-refresh, thus allowing DRAM cells to decay.

It should be noted that two key modifications are made to the default `cl_dram_dma` logic. First of all, the memory scrubber module `mem_scrb` (which erases DRAMs when the AFI is loaded) is disabled through the macro `NO_CL_TST_SCRUBBER`. This is necessary to ensure that the decay-based DRAM PUF fingerprints are not zeroed out before they are read. And, second, the error correction logic is also turned off by setting the `ECC` parameter of the `ddr4_core_ddr4` module to `OFF`. This ensures that the PUF response remains usable by keeping decay-based errors intact, instead of being corrected by ECC. The ability to turn off ECC is discussed further in Section 5.

3.2 Collecting DRAM PUF Fingerprints

Figure 3 depicts the data collection process of the decay-based DRAM PUFs. As Figure 3 shows, there are three steps in the measurement process, which use two separate AFIs:

(1) The first step is to *write* all 1s to a fixed area within a DRAM chip. It uses `AFI-0`, which is based on the AWS example design `cl_dram_dma`, with modifications to the memory scrubber and ECC, as explained above.
(2) The second step is to *wait* for DRAM cells to decay by using *AFI-1*. This step loads an FPGA image which stops self-refresh of DRAMs A, B, and D for the chosen decay period, idling the FPGA. The `cl_hello_world` design is used for this purpose, as it does not instantiate memory controllers.

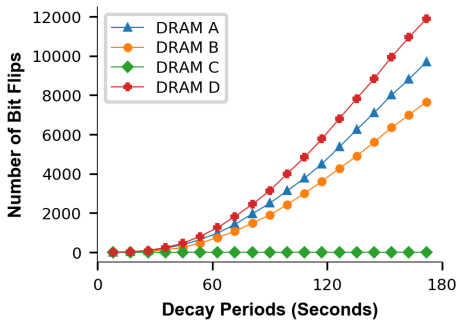

Figure 4: Number of bit flips in the four DRAMs of an FPGA board for different decay periods. DRAM C is reserved by the FPGA shell and cannot be used for PUFs. The other three DRAM error counts follow a similar pattern, but the absolute magnitudes vary.

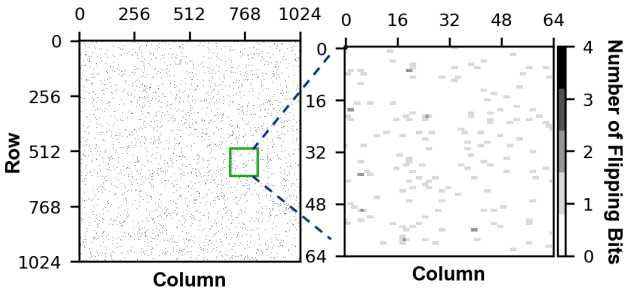

Figure 5: Bitmap of an example DRAM PUF response on an AWS FPGA, where each pixel denotes the number of bit flips per four bits in the DRAM PUF response.

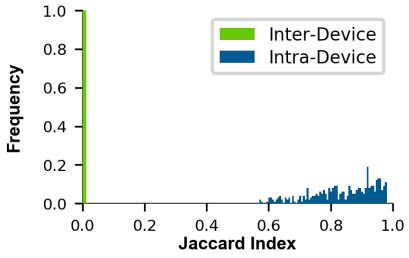

Figure 6: Distribution of Jaccard indices for each pair of DRAM PUF responses on `f1.2xlarge` instances.

(3) The final step of *reading* returns to `AFI-0`, and simply reads back the DRAM data to generate the PUF fingerprints from DRAMs A, B, and D.

4 EVALUATION

This section expands on the experimental setup (Section 4.1), and provides an example of the DRAM PUF response (Section 4.2). It then details the metric used for fingerprinting FPGA instances (Section 4.3), and calculates the probability of re-renting the same FPGA (Section 4.4). Finally, it explains that there is no overlap in the different instance types (Section 4.5), and finishes with an investigation of the background data center conditions (Section 4.6).

4.1 Data Collection on AWS

Experiments are performed on Amazon EC2 F1 spot instances [10], in the North Virginia `us-east-1` region. Spot instances are similar to on-demand ones, but can be terminated at a moment's notice. As a result, they are cheaper: an on-demand `f1.16xlarge` instance costs $13.20 per hour, while the same spot instance only $3.96 [6].

The VMs used on the cloud servers, also called Amazon Machine Images (AMIs) [8], run CentOS 7.6.1810, and access the Xilinx Virtex UltraScale+ FPGAs in the `f1` instances. A series of spot instances, launched with the same AMIs, are requested in order and are terminated after collecting DRAM PUFs responses on all FPGA slots of each instance. The interval between terminating one instance and requesting the next one is five minutes. However, due to variations in how long initialization of the FPGAs takes, there are some small differences in the collection time of the DRAM PUFs in practice. On multi-FPGA (`4x` and `16x`) instances, the measurements on different FPGA slots are done in sequence, minimizing contention errors or delays due to the shared PCIe bus.

4.2 DRAM PUF Example on Cloud FPGAs

As discussed in Sections 2 and 3, the location of bit flips which occur after disabling the memory scrubber, error correction, and self-refresh is related to the manufacturing process and can fingerprint the DRAM modules attached to the FPGAs. It can thus serve as a proxy for fingerprinting the cloud FPGA instances, under the reasonable assumption that the same DRAM chips are always permanently and physically connected to the same FPGA board. Figure 4 shows the number of bit flips (error counts) for the four DRAMs on an FPGA board after waiting for different decay periods. Due to the influence of memory access on DRAM PUFs, all data points in Figure 4 are independent. The waiting time between measurements is two minutes, and the size of the PUF is 512 kB. Decay on DRAM C cannot be measured, as it is reserved by the shell, but the other three DRAMs follow a similar pattern: the longer the wait, the more pronounced the decay. However, the absolute magnitude varies due to manufacturing variations. The decay period is chosen as 120 seconds in the following experiments.

Figure 5 shows an example DRAM PUF response, with each pixel in the 1024×1024 grid representing four bits in the 512 kB PUF response. There is sufficient randomness in the response to distinguish between otherwise-identical DRAMs.

4.3 Fingerprinting Metric

To quantify how similar or different DRAM PUF responses are, we use the Jaccard index [21]. Let F_1 and F_2 denote the set of bit flips in two DRAM PUF responses. Then, the Jaccard index for the two DRAM PUF responses is defined as:

$$J(F_1, F_2) = \frac{|F_1 \cap F_2|}{|F_1 \cup F_2|} \quad (1)$$

As shown by Xiong et al. [43], the *intra-device* Jaccard index J of PUF responses from the same DRAM chip is close to one, whereas the *inter-device* Jaccard index J from different DRAMs is close to zero. This remains true for the data collected in our work, where

Table 1: Number and type of FPGA instances rented, along with the number of unique sets of FPGAs found and the approximate experimental cost using spot instances.

F1 Type	Number of FPGAs	Unique FPGAs	Approx. Cost ($)
2xlarge	60 × 1	10 × 1	3.47
4xlarge	60 × 2	6 × 2	8.91
16xlarge	60 × 8	8 × 8	83.16

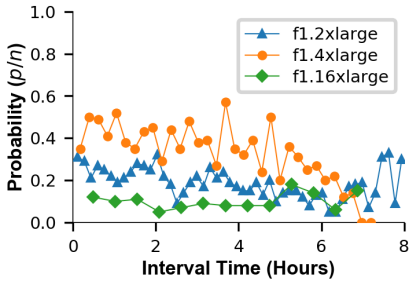

Figure 7: Probability of renting re-allocated FPGA boards for all three instance types and different waiting periods. Although the figure only shows slot 0, the probability for all slots is identical, as FPGA ordering does not change within instances.

sixty `f1.2xlarge` instances are launched in series. Due to the AWS allocation process, these instances may or may not use different FPGA boards. As shown in Figure 6, the distribution of the Jaccard indices for each pair of PUF responses has a peak close to 0, and the rest are between 0.5 and 1 as expected. Therefore, DRAM PUF responses which have a Jaccard index of less (resp. more) than 0.5 are assumed to come from different (resp. the same) FPGA boards.

4.4 Identifying Repeated Instances

This section identifies the number of unique FPGAs when renting `f1.2xlarge`, `f1.4xlarge`, `f1.16xlarge` instances sixty times each. As these instance types contain 1, 2, and 8 FPGA boards respectively, DRAM PUF fingerprints are measured on a total of 60 + 120 + 480 = 660 FPGAs. Table 1 summarizes the number of unique FPGAs seen on AWS, as indicated by the Jaccard indices of their DRAM PUFs. The results indicate that only 10, 6, and 8 unique FPGA sets have been allocated for each instance type.

Given that we observed the same FPGA multiple times, Figure 7 plots the probability of getting the same FPGA board in the North Virginia region, as a function of the amount of time between requests for two instances. As DRAM PUFs are collected in sequence, the intervals between two adjacent measurements are nearly identical. For a given time period t, all n pairs of measurements that are (approximately) t minutes apart are used to calculate the probability p/n of renting a re-allocated FPGA, where p denotes the number of pairs (out of n) for which Jaccard indices are bigger than 0.5. As n varies for different interval times, $n = 59$ for the first data point of Figure 7, and reduces by 1 for each data point to 10, 27, and 47 for 2x, 4x, and 16x instances respectively.

Although the probability appears random and hard to predict, it is non-zero for all instance types most of the time, and often close to 25% to 30% for 2x and 4x instances. As a result, temporal covert

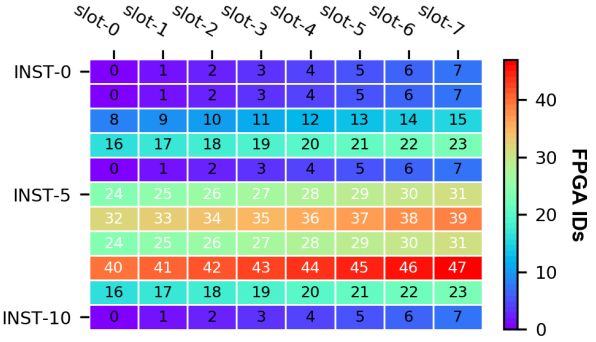

Figure 8: Fingerprinting FPGAs on `f1.16xlarge` instances with 8 FPGA slots: out of 11 spot instances, only 6 different sets of FPGAs are allocated. In the remaining instances, only 2 additional sets were identified (Table 1).

channels [38] indeed seem possible: the attacker and the victim end up on the same FPGA in consecutive time slots after about four tries on average. For 16x instances, the probability is around 10%, requiring about ten tries to get the same instance.

Figure 8, in particular, shows the results of renting 16x instances eleven consecutive times. As can be seen in the figure, one set of FPGAs is repeated four times, two are repeated two times, while three are allocated once. Moreover, the same eight FPGAs are re-allocated at once: in other words, by identifying that, e.g., DRAM D on slot 0 has stayed the same in INST-0 and INST-1, an adversary is able to deduce that all eight FPGAs have stayed the same.

4.5 (No) Overlap in Instance Types

We also investigate whether FPGAs are reused between 2x, 4x, and 16x instances, which contain 1, 2, and 8 FPGA boards respectively. However, the Jaccard index between PUFs from different instance types is close to 0, indicating that FPGAs are not shared among them. This lack of overlap can frustrate adversarial attempts at being co-located with a victim circuit on a nearby FPGA on the same server rack. In other words, although attackers can identify which FPGA they have rented, they cannot deduce the physical proximity to each other. That said, by monitoring the background conditions of the data center (Section 4.6), an adversary might still get some information about whether two FPGAs are nearby.

4.6 Monitoring Temperature Changes

We finally also investigate whether one can infer patterns about the environmental conditions of the data center in which we performed measurements. To that end, we measure how the DRAM decay varies in a span of approximately three days. As Figure 9 reveals, the PUF behaves differently throughout the measurement period. As DRAM decay varies with temperature [42], these variations can give insights into the workloads and operating conditions of the servers. For example, there may be a decrease in activity at certain times in the day, allowing the data center to cool, and the DRAM PUF to result in fewer errors. An attacker might use these insights to reason about data center capacity, and launch attacks on server availability [15, 19, 20].

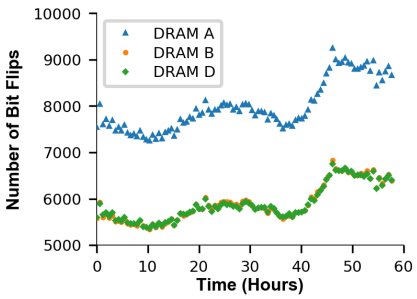

Figure 9: DRAM decay measured in the course of three days can reveal information about the data center environmental conditions. The experiment was done on a spot `f1.2xlarge` instance.

5 DEFENSE STRATEGIES

In this section, we propose several countermeasures to prevent the adversaries from being able to fingerprint cloud FPGAs.

First of all, DRAM PUFs are possible because AWS currently retains DRAM data even if the FPGA has been cleared, or an image without a memory controller is loaded. In other words, although "DRAM Data retention is not supported for CL designs with less than 4 DDRs enabled" [9], the DRAM data is not erased. Consequently, clearing or refreshing the DRAM in either of these two cases would prevent our fingerprinting approach. At the same time, it would still allow the intended use-case of the data retention feature, namely sharing data between consecutively-loaded AFIs.

Second, we disabled ECC to reliably identify the locations of bit flips and measure the response of the DRAM PUF. Disabling ECC could be banned, but at a cost of energy usage for designs that don't need it. Moreover, ECC is not guaranteed to entirely prevent our fingerprints. For example, researchers have shown that attacks using DRAM are possible even with error correction enabled [13].

Furthermore, introducing randomness at different layers of abstraction can raise the bar for the adversaries. Currently, our work can identify all eight FPGAs in an `f1.16xlarge` instance by measuring the PUF behavior on a single DRAM module (e.g., DRAM D) on one FPGA. However, software can randomize the order of FPGAs within an instance as they appear to the user, or the way the DRAM modules are presented to the FPGA. Moreover, DRAM address scrambling in the memory controller can prevent the DRAM PUF from operating. However, other types of PUFs, such as those using ring oscillators (ROs), may still be effective in fingerprinting. Although AWS prohibits traditional RO designs, alternative ROs have been proposed which could be deployed on cloud FPGAs [16, 32].

Finally, Amazon's practice of not sharing hardware between different instance types (e.g., `f1.2xlarge` and `f1.16xlarge` instances) is a good way to make finding co-located FPGAs on the same server rack more challenging. For similar reasons, it would be useful to enable remote access to FPGAs over RDMA-like protocols [24], or by dynamically attaching FPGAs to a given VM instance, as is currently possible with GPUs on AWS [7].

Although these approaches make power-based attacks harder, they cannot eliminate temporal thermal channels (e.g., [38]). As such attacks only exploit temperature effects, a mandatory cool-down period before re-assigning FPGAs can prevent covert channels, even if adversaries successfully fingerprint the devices.

6 RELATED WORK

Recent research on cloud FPGAs has shown that they are susceptible to covert-channel attacks between different designs that are simultaneously deployed (*multi-tenant attacks*) or that are consecutively instantiated (*temporal attacks*) on the same FPGA board. In the former category, co-located routing resources ("long wires") have been shown to leak information about their state to nearby (but independent) long wires in Amazon and Huawei FPGAs [16]. Similarly, a high-bandwidth covert channel can be established between logic that is physically isolated onto separate dies of the same FPGA chip (Super Logic Regions or SLRs) on the same cloud platforms [17]. In the latter category, Tian and Szefer have demonstrated a temporal thermal covert channel on Microsoft Catapult servers [38]. These three attacks depend on ring oscillators as receivers, but as Amazon prohibits combinatorial loops [9], alternative designs can be used that bypass such checks [16, 32]. In general, detecting ring oscillators and time-to-digital converters can protect against many types of remote FPGA attacks [23], but not the DRAM PUF fingerprinting approach we introduced in this paper. PUFs [44] and other designs [14] can also be used for the protection of Intellectual Property (IP) cores. However, to the best of our knowledge, PUFs have not been used for fingerprinting of cloud FPGAs in the past, despite the fact that they enable several types of applications [18, 39]. It should be noted that besides the decay-based PUF [27–30, 33, 43] used in this paper, the latency [22, 34] and startup values [35] of DRAM can also be used due to their unique characteristics. However, they cannot be deployed on cloud servers due to limitations with the APIs provided by the shell.

7 CONCLUSION

This paper focused on how to deduce aspects of the cloud FPGA infrastructure itself, without attacking it or damaging it. It introduced a novel algorithm for fingerprinting cloud FPGAs through decay-based DRAM PUFs. These PUFs made use of unintentional properties of the cross-AFI data sharing feature to bypass restrictions on the refresh parameters of the memory controller. The PUFs created resulted in unique and stable fingerprints of FPGAs on all three FPGA instance types currently available on AWS (`f1.2xlarge`, `f1.4xlarge`, and `f1.16xlarge`). Although we identified repeated FPGA allocations in our experiments within each instance type, we found no evidence of overlaps between different instance types even when requesting FPGAs a few minutes apart. Finally, we discussed defense mechanisms to protect against cloud FPGA fingerprinting. Overall, our work highlights a need for even tighter controls of the underlying hardware resources to prevent identification of the physical infrastructure as well as related attacks.

ACKNOWLEDGEMENT

This work was supported by NSF grants 1901901 and 1651945, and by Semiconductor Research Corporation (SRC) award number 2844.001. We would like to thank Amazon for donation of AWS Cloud Credits for Research.

REFERENCES

[1] Alibaba Cloud. 2019. Elastic Compute Service: Instance Type Families. https://www.alibabacloud.com/help/doc-detail/25378.htm. Accessed: 2019-09-14.

[2] Amazon Web Services. 2016. Developer Preview – EC2 Instances (F1) with Programmable Hardware. https://aws.amazon.com/blogs/aws/developer-preview-ec2-instances-f1-with-programmable-hardware/. Accessed: 2019-09-14.

[3] Amazon Web Services. 2018. Amazon EC2 F1 Instances Adds New Features and Performance Improvements. https://aws.amazon.com/about-aws/whats-new/2018/07/amazon-ec2-f1-instances-adds-new-features-and-performance-improvements/. Accessed: 2019-09-14.

[4] Amazon Web Services. 2018. AWS Shell Interface Specification. https://github.com/aws/aws-fpga/blob/master/hdk/docs/AWS_Shell_Interface_Specification.md. Accessed: 2019-09-14.

[5] Amazon Web Services. 2019. Amazon EC2 F1 Instance Partners. https://aws.amazon.com/ec2/instance-types/f1/partners/. Accessed: 2019-09-14.

[6] Amazon Web Services. 2019. Amazon EC2 Spot Instances Pricing. https://aws.amazon.com/ec2/spot/pricing/. Accessed: 2019-09-14.

[7] Amazon Web Services. 2019. Amazon Elastic Graphics. https://aws.amazon.com/ec2/elastic-graphics/. Accessed: 2019-09-14.

[8] Amazon Web Services. 2019. Amazon Machine Images (AMI). https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html. Accessed: 2019-09-14.

[9] Amazon Web Services. 2019. AWS EC2 FPGA HDK+SDK Errata. https://github.com/aws/aws-fpga/blob/master/ERRATA.md. Accessed: 2019-09-14.

[10] Amazon Web Services. 2019. AWS EC2 Spot Instances. https://aws.amazon.com/ec2/spot/. Accessed: 2019-09-14.

[11] Amazon Web Services. 2019. Official Repository of the AWS EC2 FPGA Hardware and Software Development Kit. https://github.com/aws/aws-fpga. Accessed: 2019-09-14.

[12] Baidu Cloud. 2019. FPGA Cloud Compute. https://cloud.baidu.com/product/fpga.html. Accessed: 2019-09-14.

[13] Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, and Herbert Bos. 2019. Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks. In *IEEE Symposium on Security and Privacy (S&P)*.

[14] Muhammad E. S. Elrabaa, Mohamed A. Al-Asli, and Marwan H. Abu-Amara. 2019. A Protection and Pay-per-use Licensing Scheme for On-cloud FPGA Circuit IPs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 12, 3 (Aug. 2019), 13:1–13:19.

[15] Xing Gao, Zhang Xu, Haining Wang, Li Li, and Xiaorui Wang. 2018. Reduced Cooling Redundancy: A New Security Vulnerability in a Hot Data Center. In *Network and Distributed Systems Security Symposium (NDSS)*.

[16] Ilias Giechaskiel, Kasper B. Rasmussen, and Jakub Szefer. 2019. Measuring Long Wire Leakage with Ring Oscillators in Cloud FPGAs. In *International Conference on Field Programmable Logic and Applications (FPL)*.

[17] Ilias Giechaskiel, Kasper B. Rasmussen, and Jakub Szefer. 2019. Reading Between the Dies: Cross-SLR Covert Channels on Multi-Tenant Cloud FPGAs. In *IEEE International Conference on Computer Design (ICCD)*.

[18] Jorge Guajardo, Sandeep S. Kumar, Geert-Jan Schrijen, and Pim Tuyls. 2008. Brand and IP Protection with Physical Unclonable Functions. In *IEEE International Symposium on Circuits and Systems (ISCAS)*.

[19] Mohammad A. Islam and Shaolei Ren. 2018. Ohm's Law in Data Centers: A Voltage Side Channel for Timing Power Attacks. In *ACM Conference on Computer and Communications Security (CCS)*.

[20] Mohammad A. Islam, Shaolei Ren, and Adam Wierman. 2017. Exploiting a Thermal Side Channel for Power Attacks in Multi-Tenant Data Centers. In *ACM Conference on Computer and Communications Security (CCS)*.

[21] Paul Jaccard. 1901. Étude Comparative de la Distribution Florale dans une Portion des Alpes et du Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles* 37 (1901), 547–579.

[22] Jeremie S. Kim, Minesh Patel, Hasan Hassan, and Onur Mutlu. 2018. The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*.

[23] Jonas Krautter, Dennis R. E. Gnad, and Mehdi B. Tahoori. 2019. Mitigating Electrical-level Attacks Towards Secure Multi-Tenant FPGAs in the Cloud. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 12, 3 (Aug. 2019), 12:1–12:26.

[24] Joshua Lant, Andrew Attwood, Javier Navaridas, Mikel Lujan, and John Goodacre. 2019. Receive-Side Notification for Enhanced RDMA in FPGA Based Networks. In *International Conference on Architecture of Computing Systems (ARCS)*.

[25] Microsoft Research. 2017. Microsoft Unveils Project Brainwave for Real-time AI. https://www.microsoft.com/en-us/research/blog/microsoft-unveils-project-brainwave/. Accessed: 2019-09-14.

[26] Nimbix, Inc. 2019. Xilinx Alveo Accelerator Cards. https://www.nimbix.net/alveo/. Accessed: 2019-09-14.

[27] Amir Rahmati, Matthew Hicks, Daniel E. Holcomb, and Kevin Fu. 2015. Probable Cause: The Deanonymizing Effects of Approximate DRAM. In *Annual International Symposium on Computer Architecture (ISCA)*.

[28] Sami Rosenblatt, Srivatsan Chellappa, Albert Cestero, Norman Robson, Toshiaki Kirihata, and Srikanth S. Iyer. 2013. A Self-Authenticating Chip Architecture Using an Intrinsic Fingerprint of Embedded DRAM. *IEEE Journal of Solid-State Circuits (JSSC)* 48, 11 (Nov. 2013), 2934–2943.

[29] Sami Rosenblatt, Daniel Fainstein, Albert Cestero, John Safran, Norman Robson, Toshiaki Kirihata, and Srikanth S. Iyer. 2013. Field Tolerant Dynamic Intrinsic Chip ID Using 32 nm High-K/Metal Gate SOI Embedded DRAM. *IEEE Journal of Solid-State Circuits (JSSC)* 48, 4 (April 2013), 940–947.

[30] André Schaller, Wenjie Xiong, Nikolaos A. Anagnostopoulos, Muhammad U. Saleem, Sebastian Gabmeyer, Stefan Katzenbeisser, and Jakub Szefer. 2017. Intrinsic Rowhammer PUFs: Leveraging the Rowhammer Effect for Improved Security. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*.

[31] André Schaller, Wenjie Xiong, Nikolaos A. Anagnostopoulos, Muhammad U. Saleem, Sebastian Gabmeyer, Boris Skoric, Stefan Katzenbeisser, and Jakub Szefer. 2018. Decay-Based DRAM PUFs in Commodity Devices. *IEEE Transactions on Dependable and Secure Computing (TDSC)* 16, 3 (May 2018), 462–475.

[32] Takeshi Sugawara, Kazuo Sakiyama, Shoei Nashimoto, Daisuke Suzuki, and Tomoyuki Nagatsuka. 2019. Oscillator without a Combinatorial Loop and its Threat to FPGA in Data Centre. *Electronics Letters* 55, 11 (May 2019), 640–642.

[33] Soubhagya Sutar, Arnab Raha, and Vijay Raghunathan. 2016. D-PUF: An Intrinsically Reconfigurable DRAM PUF for Device Authentication in Embedded Systems. In *International Conference on Compliers, Architectures, and Sythesis of Embedded Systems (CASES)*.

[34] B. M. S. Bahar Talukder, Biswajit Ray, Domenic Forte, and Md Tauhidur Rahman. 2018. PreLatPUF: Exploiting DRAM Latency Variations for Generating Robust Device Signatures. *IEEE Access* 7 (June 2018), 81106–81120.

[35] Fatemeh Tehranipoor, Nima Karimian, Kan Xiao, and John Chandy. 2015. DRAM-Based Intrinsic Physically Unclonable Functions for System-Level Security and Authentication. In *Great Lakes Symposium on VLSI (GLSVLSI)*.

[36] Tencent Cloud. 2019. Cloud Virtual Machine Instance Types. https://intl.cloud.tencent.com/document/product/213/11518. Accessed: 2019-09-14.

[37] Texas Advanced Computing Center. 2015. TACC to Launch New Catapult System to Researchers Worldwide. https://www.tacc.utexas.edu/-/tacc-to-launch-new-catapult-system-to-researchers-worldwide. Accessed: 2019-09-14.

[38] Shanquan Tian and Jakub Szefer. 2019. Temporal Thermal Covert Channels in Cloud FPGAs. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*.

[39] Pim Tuyls, Geert-Jan Schrijen, Frans Willems, Tanya Ignatenko, and Boris Skoric. 2007. Secure Key Storage with PUFs. In *Security with Noisy Data: On Private Biometrics, Secure Key Storage and Anti-Counterfeiting*, Pim Tuyls, Boris Skoric, and Tom Kevenaar (Eds.). Springer, Chapter 15, 269–292.

[40] Xilinx, Inc. 2017. Xilinx Powers Huawei FPGA Accelerated Cloud Server. https://www.xilinx.com/news/press/2017/xilinx-powers-huawei-fpga-accelerated-cloud-server.html. Accessed: 2019-09-14.

[41] Xilinx, Inc. 2019. UltraScale Architecture Configuration: User Guide (UG570). https://www.xilinx.com/support/documentation/user_guides/ug570-ultrascale-configuration.pdf. Accessed: 2019-09-14.

[42] Wenjie Xiong, Nikolaos A. Anagnostopoulos, André Schaller, Stefan Katzenbeisser, and Jakub Szefer. 2019. Spying on Temperature using DRAM. In *Design, Automation, and Test in Europe (DATE)*.

[43] Wenjie Xiong, André Schaller, Nikolaos A. Anagnostopoulos, Muhammad U. Saleem, Sebastian Gabmeyer, Stefan Katzenbeisser, and Jakub Szefer. 2016. Runtime Accessible DRAM PUFs in Commodity Devices. In *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*.

[44] Jiliang Zhang and Gang Qu. 2019. Recent Attacks and Defenses on FPGA-based Systems. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 12, 3 (Aug. 2019), 14:1–14:24.